Software Architecture of the Light Weight Kernel, Catamount

May 19, 2005
Sue Kelly
Sandia National Laboratories
smkelly@sandia.gov, 505-845-9770

SAND-2005-2781C
SUNMOS, PUMA, Cougar, Catamount Design Goals

- Targeted at massively parallel environments comprised of thousands of processors with distributed memory and a tightly coupled network.
- Provide necessary support for scalable, performance-oriented scientific applications.
- Offer a suitable development environment for parallel applications and libraries.
- Emphasize efficiency over functionality.
- Maximize the amount of resources (e.g. CPU, memory, and network bandwidth) allocated to the application.
- Seek to minimize time to completion for the application.
Catamount is designed for an MPP environment with functional partitions.

Compute Processors (Catamount)

I/O processors (Linux)

Service Processors (Linux)

Network I/O Processors (Linux)

High Speed External Network
### Catamount Physical Memory layout

<table>
<thead>
<tr>
<th>QK text</th>
<th>QK data</th>
<th>Stack</th>
<th>Network buffer (qk heap)</th>
<th>Portals memory</th>
<th>PCT text</th>
<th>PCT data</th>
<th>PCT heap</th>
</tr>
</thead>
</table>

- **User program text & data**
- **Up to 4 instances**

Note: not to scale
Quintessential Kernel (QK)

- Policy enforcer
- Initializes hardware
- Handles interrupts and exceptions
- Maintains hardware virtual addressing
- No virtual memory support
- Static size
- Non-blocking
- Few, well-defined entry points
Process Control Thread (PCT)

• Runs in user space
• More privileged than user applications
• Policy maker
  – Process loading (with yod)
  – Process scheduling
  – Virtual address space management
  – Fault handling
  – Signals
Catamount’s libc is pruned version of glibc

- No threads support
- No off-node communication other than via Portals, such as pipes, sockets, rpc's or Internet Protocols
- No dynamic process creation; for example: no exec(), fork(), popen(), or system()
- No dynamic loading of executable code
- Limited signals support
- No /proc or ptrace
- No mmap. A skeleton function is supplied, but returns –1.
- No profil()
- Limited ioctl
- No getpwd family of calls
- No functions requirement any form of db (e.g. ndb). For example, there is no support for the uid, gid family of queries that based on the ndb.
- No terminal control
- No functions that require UNIX-style daemons
- Custom catamount malloc is used by default
Libsysio routes I/O calls to the appropriate file system handler
Libcatamount

- RPC mechanism to communicate with yod for stdio and system call offload
- Custom malloc tuned for large allocations
- Pre-main initialization
- Interface routines for PCT and QK services
Libportals

- Message passing API
- Separate software package
- Required by Catamount
- http://www.sourceforge.net/packages/sandiaportals
YOD runs in the service partition

• Functions
  – Controls the logarithmic launch of a parallel job
  – Proxies standard I/O, plus other I/O, if necessary
  – Manages the parallel job throughout its run

• Yod is an evolution of the xnc (eXecute Network Computer) program used to launch jobs on the nCube: \((x+1)(n+1)(c+1) = yod\)

• `yod [ -Account project task ] [ -D option ] [ -help ] [{ -size | -sz | -np }{ n | all }] [ -stack size ] [ -tlimit secs ] [ -list processor-list ] [-strace] [-target { catamount | linux }] [ -share ] [ -heap size ] [ -Priority priority ] [-Version] progrname [ progargs ] | -F loadfile

YOD runs in the service partition
Multi-Partition Job Support is new with Catamount

• Support for parallel applications that span Catamount and Linux
  – Yod using load file option (-F)
  – Requires a PCT to run on Linux
  – Requires different executables
  – Creates one MPI_COMM_WORLD
Future Plans

• Re-introducing support for dual processors that removed in the port from cougar to catamount
• Studying whether catamount is viable for multi-core (> 2 CPUs) support